Fuzzy rules from ant-inspired computation

Shen, Qiang; Galea, Michelle

Publication date:
2004

Citation for published version (APA):
Shen, Q., & Galea, M. (2004). *Fuzzy rules from ant-inspired computation*. 1691-1696.
http://hdl.handle.net/2160/452

25–29 July, 2004 • Budapest, Hungary

Fuzzy Rules from Ant-Inspired Computation

Michelle Galea and Qiang Shen
School of Informatics
University of Edinburgh, U.K.
{m.galea@sms,qiangs@inf}.ed.ac.uk

Abstract— **A new approach to fuzzy rule induction from historical data is presented. The implemented system - *FRANTIC* - is tested on a simple classification problem against a fuzzy tree induction algorithm, a genetic algorithm, and a numerical method for inducing fuzzy rules based on fuzzy subsethood values. The results obtained by *FRANTIC* indicate comparable or better classification accuracy, superior comprehensibility, and potentially more flexibility when applied to larger data sets. The impact of the knowledge representation used when generating fuzzy rules is also highlighted.**

I. Introduction

The activities of social insects have inspired successful applications in many areas. For instance, the cemetery organisation and brood sorting activity of ants has led to new clustering algorithms (e.g. graph colouring and partitioning [1]), while their foraging behaviour has resulted in a suite of optimisation algorithms - called ant algorithms - for solving problems such as bin packing and the travelling salesman problem [2]. It is these optimisation algorithms that have been adapted for rule induction.

The appeal of ant algorithms lies in several factors: they provide a simple effective mechanism for conducting global search by simultaneously constructing multiple solutions that investigate diverse areas of the solution space; a simplicity of implementation that requires minimum understanding of the problem domain; the problem-specific elements - such as the fitness function and heuristic - which may be readily borrowed from existing literature on rule induction; and, an explicit heuristic embedded in the solution construction mechanism that makes for easy insertion of domain knowledge.

The application of ant algorithms to rule induction is an unexplored research area. This paper develops work done in [3] for inducing crisp rules, modifying the overall strategy used and extending it for the induction of linguistic fuzzy IF-THEN rules. The implementation is tested on a classification problem, but this approach is equally applicable to fuzzy modelling for other tasks, e.g. prediction.

In the next section the motivation and details behind the Ant Colony Optimisation (ACO) metaheuristic, instantiations of which result in different ant algorithms, are introduced. The existing literature on the application of ACO-based algorithms to rule induction is also briefly reviewed. In Section III the *FRANTIC* system is described, while Section IV presents the initial findings of the system, comparing it with the results obtained from several other fuzzy rule induction algorithms. Section V highlights the advantages and limitations of the current research, which in turn suggest avenues for future work.

II. Ant Algorithms

Ant algorithms is a collective term for algorithms motivated by the way ants forage for food. Experiments with real ant colonies have been conducted to determine how ants are able to find the shortest path between their nest and a food source. In making decisions about which path to take ants are guided by the amount of pheromone (a chemical substance laid by ants) on a path - the greater the amount the higher the probability an ant will follow that path.

When a new food source is first located there is no pheromone to guide ants and so each will have made a random decision when presented with different paths. Several paths by different ants may therefore have been taken to reach the same food source. On their return trip to the nest ants will lay more pheromone and those that have found the shortest path will get back to the nest more quickly. Pheromone, however, evaporates so unless the amount on a path is maintained by ants that continue to use it, the path will be chosen less frequently. The shortest path will however continue to accrue more pheromone as ants are able to travel faster along this path and can lay pheromone more quickly to replace that which is evaporating. It has been observed that foraging ants usually converge on the shortest path to a food source.

A. Ant Colony Optimisation

ACO is an agent-based meta-heuristic motivated by these foraging strategies of real ants. In ACO, each artificial ant is considered a simple agent, communicating with other ants indirectly by effecting changes to a common environment. A high-level description of an ACO-based algorithm is:

```
(1)  while termination condition false
(2)     each ant constructs a new solution
(3)     evaluate new solutions
(4)     update pheromone levels
(5)  output best solution
```

The pheromone levels and a problem-specific heuristic are what guide artificial ants in their construction of a solution. Any combinatorial problem for which the following listed elements can be defined may be solved by an ACO algorithm [4]. These elements are introduced briefly here in the context of rule induction, with more detail provided in Section III. The first four elements relate to line (2) of the high-level ACO

0-7803-8353-2/04/$20.00 © 2004 IEEE

description above, the fifth relates to line (3), and the sixth to line (4):

1) An appropriate *problem representation* is required that allows an artificial ant to incrementally build a solution using a *probabilistic transition rule*. The main idea is to model the problem as the search for a best path through a graph. In the context of rule induction a solution is a rule antecedent and each node of the graph represents a condition that may form part of it, such as OUTLOOK=Sunny, or OUTLOOK=Cloudy.
2) A local *heuristic* provides guidance to an ant in choosing the next node for the path (solution) it is building. Possible examples may be based on fuzzy subsethood values, or a measure of the vagueness in a fuzzy set.
3) The *probabilistic transition rule* determines which node an ant should visit next. The transition rule is dependent on the *heuristic* value and the *pheromone* level associated with a node.
4) A *constraint satisfaction method* forces the construction of feasible rules. For instance, if simple propositional IF-THEN rule antecedents are being constructed, then only one fuzzy linguistic term from each fuzzy variable may be selected.
5) A *fitness function* determines the quality of the solution built by an ant.
6) The *pheromone update rule* specifies how to modify the pheromone levels of each node in the graph. For instance, between iterations of an ACO algorithm, the nodes (conditions) contained in the best rule antecedent created get their pheromone levels increased.

B. Rule Induction via Ant Colony Optimisation

The application of ant-inspired algorithms to rule induction is an unexplored research area.

A first attempt was made by Casillas *et al* in [5]. However, the ACO algorithm is not used for generating fuzzy rules, but for assigning rule conclusions. In their problem graph the fixed number of nodes are fuzzy rule antecedents found by a deterministic method from the training set. An ant goes round the problem graph, visiting each and every node in turn and probabilistically assigns a rule conclusion to each.

In [3] an ACO algorithm is used for generating crisp IF-THEN rule antecedents. In the problem graph each node represents a condition that may be selected as part of the crisp rule antecedent being built by an ant. An ant goes round the graph selecting nodes according to a constraint satisfaction method, building its rule antecedent. The rule conclusion is assigned afterwards by a deterministic method. The overall strategy used is one of iterative rule learning - starting with a full training set an ACO algorithm is run and the best rule created by an ant is added to a final rule set. Instances in the training set that are covered by this best rule are removed before a new ACO algorithm is run. This process is re-iterated until only a few (as pre-determined by the user) instances remain in the training set, when a default rule is created to cover them. The final result is an ordered rule list with the rules being applied in the order in which they were created, when classifying a new instance.

III. THE *FRANTIC* SYSTEM

FRANTIC (Fuzzy Rules from ANT-Inspired Computation) implements a class-dependent iterative rule learning stategy whereby for each class that requires descriptive rules to be learnt, a number of ant algorithms are run with each one outputting one such rule. Note that lines (4)-(8) are equivalent to the ACO-based algorithm previously introduced:

```
(1)  for each class
(2)     reinstate full training set
(3)     while classInstRem>classInstUncovered
(4)        for noIterations
(5)           each ant constructs rule
(6)           evaluate all rules
(7)           update pheromone levels
(8)        add best rule to finalRuleSet
(9)        remove covered class instances
(10) output finalRuleSet
```

A simplified version of this strategy is to run just one ACO algorithm for each class, with the assumption being that one rule is sufficient to describe a class. *FRANTIC* may be run in this simplified mode, or carry out a full class-dependent iterative strategy.

If a full version is run, then more than one ACO algorithm may be run per class. From each the best rule constructed is determined and added to the final rule set. However, before the next ACO is run to find another rule describing the same class, the instances belonging to that class that are covered by the previous best rule are removed from the training set. This process goes on until there are fewer class instances remaining in the training set than a value pre-defined by the user, line (3). This parameter provides a simple and effective mechanism for controlling over-fitting to the training data, since continuing to run ACO algorithms to find rules describing the last few class instances may not necessarily produce rule sets with better classification accuracy. Instances belonging to classes other than the one currently being described are left in the training set, as this helps in the evaluation of the rules constructed.

The following subsections explain the ACO-specific elements in more detail.

A. Rule Construction

FRANTIC has been designed with the flexibility to create simple propositional rules, propositional rules with internal disjunction (e.g. OUTLOOK=Cloudy OR Sunny), or propositional rules that include negated terms (e.g. OUTLOOK=NOT_Sunny). The problem graph for each is similar and differences in the rule construction mechanism from one knowledge representation to the other are highlighted where appropriate.

When creating a rule antecedent an ant traverses a problem graph where each node represents a term that may be added e.g OUTLOOK=Sunny. In the case of constructing rules with negated terms, the graph has double the number of nodes - one extra for each original linguistic term, e.g.

OUTLOOK=NOT_Sunny. The choice of the next node to visit (next term to be added to the current partial rule antecedent) depends on both a heuristic value and the pheromone level associated with the node. The choice is made probabilistically but is biased towards terms that have relatively higher heuristic and pheromone values.

However, after selection and before a term is added to a rule antecedent, a check is made - this ensures that the resultant rule antecedent covers a minimum number of instances from the training set (set by a parameter called `minInstPerRule`), and is another simple way of avoiding over-fitting to the training. With fuzzy sets all fuzzy rules cover all training instances, but to varying degrees, and so what constitutes coverage of an instance by a fuzzy rule needs clarifying. This is defined in subsection *B*.

For simple propositional rules, or rules with negated terms, if an ant does add a term to its rule antecedent then it will not consider other linguistic terms belonging to the same linguistic variable. For example, if the linguistic variable OUTLOOK has terms Sunny, Cloudy, Rain, and the term OUTLOOK=Sunny has just been added to the rule antecedent, then the remaining terms are not considered further. If this restriction is removed, then it is possible for ants to add more than one linguistic term from each variable, with the interpretation being of a disjunctive operator between the terms added.

1) Heuristic: The heuristic used to guide ants when selecting terms is based on fuzzy subsethood values [6], giving a degree to which one fuzzy set *A* is a subset of another fuzzy set *B*:

$$S(A,B) = \frac{M(A \bigcap B)}{M(A)} = \frac{\sum_{u \in U} Min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)}$$

where in this case u is an instance from the training set U, A represents a class label and B a term that may be added to a rule antecedent. The heuristic value of a term j (η_j) therefore gives a measurement of how important that term is in describing a specific class. Since *FRANTIC* finds descriptions for each class in turn, heuristic values for terms are calculated at the start of each major iteration, line (1), using only those instances in the training set which belong to the class under consideration.

If fuzzy rules with negated terms are being constructed, the heuristic value for the negated term is the complement of the heuristic value for the non-negated term, i.e. $\eta_{NOT\_j} = 1 - \eta_j$

2) Pheromone Updating: At the start of an ACO run, line (4), all nodes in the graph have an equal amount of pheromone which is set to the inverse of the number of nodes. The pheromone level of individual nodes, however, changes between iterations, line (7). Towards the end of each iteration, line (6), rules created by all ants are evaluated. The terms in the best rule, say *R*, then get their pheromone levels increased:

$$\tau_j(t+1) = \tau_j(t) + \tau_j(t) \cdot Q, \forall j \in R$$

i.e. at time *t+1* each term *j* in rule *R* gets its pheromone level increased in proportion to the the quality of the rule *Q*. A normalisation of pheromone levels of *all* terms further results in a decrease of the pheromone levels of terms *not* in *R*.

The pheromone updating process is a reinforcement mechanism - both positive and negative - for ants constructing new rules in successive iterations: terms that have had their pheromone levels increased have a higher chance of being selected, while those that have had their levels decreased have a lower chance.

3) Transition Rule: Ants select terms while constructing a rule antecedent according to a transition rule that is probabilistic but biased towards terms that have higher heuristic and pheromone levels. The relative importance of the heuristic and pheromone may be controlled by adjusting the values of the parameters α and β, though they have been kept constant and equal in these experiments. The probability that a term j is selected by an ant is given by:

$$P_j = \frac{[\eta_j]^\alpha \cdot [\tau_j]^\beta}{\sum_{i=1}^{n} (\eta_i \cdot \tau_i), \forall i \in I}$$

where i is any term within the set I of all terms, and n is the number of terms in the graph.

The probabilistic nature of the rule is a way of introducing exploration into the search for a solution, in the expectation that a more optimal solution may well be found rather than by adhering strictly to terms with the highest values.

B. Rule Evaluation

After a rule has been constructed, it needs to be evaluated. This is done by assessing how accurate the rule is in classifying the training instances. Before the fitness function is discussed, what constitutes coverage (or matching) of a fuzzy instance by a fuzzy rule needs to be defined. This paper adopts similar definitions for fuzzy rule matching and classification by fuzzy rules (subsection *C* below) as in [8], in order to make comparisons with the results from that work as equitable as possible.

1) Fuzzy Rule Matching: A fuzzy rule is said to cover or match a fuzzy instance if their degree of match is equal to or greater than a pre-defined value, here called a threshold value.

When rule *R* is applied to instance *u* it is necessary to determine how well the attributes of *u* match the condition part of *R*, and how the class of *u* matches the conclusion of *R*. An example follows.

Consider a rule *R*=(1,1,0; 0,0,1; 1,1; 0,1) that represents a rule with four attributes, the last being the class attribute with two possible values. Terms that are present in the rule are denoted by 1, others by 0. These rules may only classify instances into one class. However, the condition attributes may take more than one value (i.e. propositional rules with internal disjunction).

Consider a fuzzy instance *u*=(0.1,0.6,0; 0.1,0,0.8; 0.3,0.4; 0.2,0.7) where the representation is the same as for rule *R*, though the conclusion attribute values may be greater than 0 for more than one class.

The degree of match between *R* and *u* is given by

$$mRule(R,u) = Min(mCond(R,u), mConc(R,u))$$

where the degree of condition match between R and u is

$$mCond(R,u) = Min_k(mAtt_k(R,u))$$

and the degree of conclusion match is

$$mConc(R,u) = Max_{1\leq L}(Min(\mu_{Class_j}(R), \mu_{Class_j}(u)))$$

with L being the number of class labels, i.e. the number of terms for the class attribute. In the above definitions, $mAtt_k$ measures the degree of match between an attribute k in R and the corresponding attribute in u:

$$mAtt_k = Max_j(Min(\mu_{k_j}(R), \mu_{k_j}(u)))$$

where j is a term within the domain of attribute k. However, if the terms for an attribute are all present in a rule, then the corresponding attribute match is equal to 1, the interpretation being that the attribute is irrelevant. From the rule and instance examples above the attribute matches are: $mAtt_1 = 0.6$, $mAtt_2 = 0.8$, $mAtt_3 = 1$, with a condition match $mCond(R,u) = 0.6$. The conclusion match is $mConc(R,u) = 0.7$ and the resulting rule match is $mRule(R,u) = 0.6$. If the threshold value is set at 0.6 or below, then this rule is considered to cover the instance. If the threshold value is set above 0.6, then this rule is considered to not sufficiently match the instance.

2) Fitness Function: The fitness function evaluates an individual rule on the basis of how accurately it classifies all instances in the training set. It combines a measure of the sensitivity of a rule (its accuracy among instances of the same class as the rule) with a measure of the specificity of the rule (its accuracy among instances of different classes):

$$Q = \frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}$$

where

- TP (True Positives) is the number of instances covered by the rule that have the same class label as the rule
- FP (False Positives) is the number of instances covered by the rule that have a different class label from the rule
- FN (False Negatives) is the number of instances that are not covered by the rule but have the same class label as the rule, and
- TN (True Negatives) is the number of instances that are not covered by the rule and do not have the same class label as the rule.

Whether a rule covers an instance or not is determined by a threshold value, pre-defined by the user, for the degree of match between the rule and an instance.

C. Classification by Fuzzy Rules

Once a complete fuzzy rule set has been generated, it needs to be tested for classification accuracy. This is is performed in the same way as in [7]–[9]:

1) For each rule, calculate the condition match for instance u, and set the conclusion match equal to the condition match;
2) If two or more rules classify instance u into the same class, choose the highest conclusion match as the degree for that class;
3) Finally, select the class with the highest membership degree as the class for instance u.

TABLE I
COMPARISON OF ALGORITHMS

	%Accuracy	#Rules	#Terms
Fuzzy DT [9]	81.25	6.0	1.7
Fuzzy GA [8]	87.50	5.0	3.2
Fuzzy SH [7]	93.75	3.0	2.3

IV. Results and Analyses

A. The Data Set and Other Algorithms

The problem on which *FRANTIC* is tested is the fuzzified Saturday Morning data set in [9]. The data set consists of 16 instances, 4 condition attributes and 1 class attribute called PLAN:

OUTLOOK={Sunny,Cloudy,Rain}
TEMPERATURE={Hot,Cool,Mild}
HUMIDITY={Humid,Normal}
WIND={Windy,Not-Windy}
PLAN={Volleyball,Swimming,Weightlifting}

The fuzzy rule sets generated by *FRANTIC* are compared against those produced by a fuzzy decision tree algorithm [9], a fuzzy genetic algorithm [8], and a method based on fuzzy subsethood values [7]. A summary of the results produced by these algorithms - with respect to classification accuracy on the training set, the number of rules generated, and the average number of conditions in a rule antecedent - is given in Table I. Note that only the accuracy of one rule set generated by the genetic algorithm is quoted for [8], and the assumption in this paper is that it is the best ruleset obtained.

Tables II, III, and IV show the rule sets generated by these algorithms. This is useful for comparing the rules obtained using different knowledge representations. The decision tree generates simple propositional rules, the genetic algorithm generates propositional rules with internal disjunction, while the subsethood value based method generates simple propositional rules that include negated terms. Note that the final rule produced by this last method, in Table IV, has no explanatory power of its own, as it is written in terms of the other rules.

B. FRANTIC Parameters

FRANTIC parameters that require setting are listed in Table VI, together with a description and the values given in order to obtain the results reported here. Note that the concept of matching between a rule and an instance is used several times by *FRANTIC* (rows 6-8). These thresholds have been implemented separately for maximum flexibility. Initial findings suggest rule sets with greater classification accuracy are found if the threshold during rule construction is greater than the threshold during fitness evaluation and for removal of class

TABLE II

RULE SET GENERATED BY A FUZZY DECISION TREE

R1	IF TEMPERATURE is Hot AND OUTLOOK is Sunny THEN Swimming
R2	IF TEMPERATURE is Hot AND OUTLOOK is Cloudy THEN Swimming
R3	IF OUTLOOK is Rain THEN Weightlifting
R4	IF TEMPERATURE is Mild AND WIND is Windy THEN Weightlifting
R5	IF TEMPERATURE is Cool THEN Weightlifting
R6	IF TEMPERATURE is Mild AND WIND is Not-windy THEN Volleyball

TABLE III

RULE SET GENERATED BY A FUZZY GENETIC ALGORITHM

R1	IF OUTLOOK is Sunny OR Cloudy AND TEMPERATURE is Hot THEN Swimming
R2	IF OUTLOOK is Rain THEN Weightlifting
R3	IF TEMPERATURE is Mild OR Cool AND WIND is Windy THEN Weightlifting
R4	IF OUTLOOK is Cloudy OR Rain AND HUMIDITY is Humid THEN Weightlifting
R5	IF OUTLOOK is Sunny OR Cloudy AND TEMPERATURE is Mild OR Cool AND HUMIDITY is Normal AND WIND is Not-windy THEN Volleyball

instances between ACO runs. Lowering this last threshold may well act as another mechanism for avoiding over-fitting to the training data. However, further investigation is required to understand the dynamics between these three parameters and it may well be possible to merge two or three of them.

Very little parameter tuning has been done. The number of ants per iteration, and the number of iterations were based on [8], but preliminary results suggests that for this test problem at least, *FRANTIC* is able to find equally good rulesets (to those reported in Table VII) with `noAnts=15` and `noIterations=20`. A few experiments carried out with changes to the values of other parameters are reported in the following subsection. These parameters and their optimal settings also merit further investigation.

C. FRANTIC Results

Table VII compares predictive accuracy of the rule sets generated by *FRANTIC* using a simplified iterative strategy and a full iterative strategy. Each result is the accuracy on the training set, and is the average of the accuracy obtained from 30 rule sets produced by *FRANTIC*. The values for the parameters are as stated in Table VI with different values for

TABLE IV

RULE SET GENERATED BY A METHOD BASED ON SUBSETHOOD VALUES

R1	IF OUTLOOK is NOT_Rain AND HUMIDITY is Normal AND WIND is Not-windy THEN Volleyball
R2	IF OUTLOOK is NOT_Rain AND TEMPERATURE is Hot THEN Swimming
R3	IF $MF(R1) < \beta$ AND $MF(R2) < \beta$ THEN Weightlifting

TABLE V

FRANTIC RULE SETS - SIMPLIFIED (R1-R3, 93.75% ACCURACY) VS. FULL ITERATIVE RULE LEARNING (R1-R4, 87.50% ACCURACY)

R1	IF OUTLOOK is NOT_Rain AND TEMPERATURE is NOT_Cool AND HUMIDITY is Normal AND WIND is Not-windy THEN Volleyball
R2	IF OUTLOOK is NOT_Rain AND TEMPERATURE is Hot THEN Swimming
R3	IF TEMPERATURE is Hot AND WIND is Windy THEN Weightlifting
R4	IF OUTLOOK is NOT_Sunny AND TEMPERATURE is NOT_Mild THEN Weightlifting

`constructionThreshold` per row. Columns 2 and 5 show accuracy obtained when using simple propositional rules as the knowledge representation, for the simplified iterative strategy and the full iterative strategy respectively, columns 3 and 6 when using propositional rules with internal disjunction, and columns 4 and 7 when using simple propositional rules that also include negated terms of the original terms.

The number of rules generated using the simplified iterative strategy is 3 (since there are 3 classes to be described), while the number of rules generated using the full iterative strategy is 4, with the fourth extra rule generally being one describing Weightlifting. When the number of rules per class is restricted, using a more expressive knowledge representation generally produces more accurate rule sets (columns 3 and 4 vs. column 2). When this restriction is removed there is less variance in the results between the different knowledge representations, since extra rules can compensate for the limitations of the representation.

Note, though, that for these parameter settings the highest accuracy is obtained using a simplified iterative strategy. This is thought to be due to the nature of fuzzy rules and how they interact when classifying an instance. The rule set commonly produced using a full iterative strategy with negated terms is the same as the rule set generally obtained using a simplified strategy, only it has an extra rule for Weightlifting (Table V). This extra rule blurs the distinction between the classes for Volleyball and Weightlifting when classifying an instance in the data set.

Though little parameter tuning has been attempted in these early experiments, the full iterative strategy can produce rule sets with as high a degree of accuracy. For instance, with `constructionThreshold=0.6` and `fitnessThreshold=removalThreshold=0.4`, a 4-rule set with 93.75% accuracy is created. With the same values for thresholds and `classInstUncovered=0`, a 5-rule set with same accuracy is produced. With `classInstUncovered=2` the rule set has 3 rules and the same accuracy.

V. CONCLUSION

This paper has demonstrated that the application of ant algorithms for fuzzy rule induction is a viable approach. Using a simple data set as a test problem, *FRANTIC* is able to obtain rule sets that are comparable or superior to rule sets

TABLE VI
FRANTIC PARAMETERS

Parameter Name	Description	Value
noAnts	number of ants constructing a solution within an iteration, (line (5))	100
noIterations	number of iterations per ACO run, (line (4))	25
minInstPerRule	required during rule construction - minimum number instances in training set that rule must cover (section III.A)	4
classInstUncovered	maximum number of class instances that may be left uncovered by a rule, before descriptions for a new class are found, (line (3))	1
constructionThreshold	used during construction of a rule - sets the value for the threshold below which a rule is considered to not cover an instance in the training set	various
fitnessThreshold	used during evaluation of a rule - sets the value for the threshold below which a rule is considered to not cover an instance in the training set	0.5
removalThreshold	used during removal of class instances from the training set between ACO runs, (line 9) - sets the value for the threshold below which a rule is considered to not cover an instance in the training set	0.5

TABLE VII
FRANTIC PREDICTIVE ACCURACY - SIMPLIFIED VS. FULL ITERATIVE RULE LEARNING

Construction Threshold	Simplified Iterative Rule Learning			Full Iterative Rule Learning		
	Prop % (+/-)	Pid % (+/-)	Not % (+/-)	Prop % (+/-)	Pid % (+/-)	Not % (+/-)
0.45	56.25 (0.00)	79.79 (3.15)	71.04 (8.76)	81.25 (0.00)	79.79 (2.69)	79.17 (3.00)
0.50	56.25 (0.00)	79.58 (3.99)	70.21 (8.32)	81.25 (0.00)	80.00 (3.03)	78.54 (3.15)
0.55	75.00 (0.00)	80.42 (2.98)	87.71 (5.80)	75.00 (0.00)	78.54 (4.24)	87.50 (0.00)
0.60	75.00 (0.00)	81.04 (3.15)	89.58 (3.00)	75.00 (0.00)	80.00 (2.54)	87.50 (0.00)
0.65	81.25 (0.00)	78.75 (4.95)	93.75 (0.00)	81.25 (0.00)	81.25 (0.00)	87.50 (0.00)
0.70	81.25 (0.00)	77.92 (5.57)	93.75 (0.00)	81.25 (0.00)	81.25 (0.00)	87.50 (0.00)
0.75	56.25 (0.00)	56.04 (1.14)	58.96 (3.15)	25.00 (0.00)	31.25 (0.00)	62.50 (0.00)
0.80	56.25 (0.00)	55.42 (2.16)	59.58 (3.17)	25.00 (0.00)	31.25 (0.00)	62.50 (0.00)

obtained by alternative approaches, in terms of the predictive accuracy and comprehensibility. All rules have explanatory power of their own, i.e. they are not dependent on other rules to be able to classify instances correctly. Furthermore, no assumption is made that one rule per class is sufficient to describe it, which may be detrimental on larger data sets. The simplicity of implementation for obtaining different knowledge representations has also been illustrated.

However, much work remains to be done. In line with the algorithms against which *FRANTIC* was compared, the results on predictive accuracy were obtained using the training set. *FRANTIC* now needs to be tested on larger and more complex real-world data sets, using a separate test set to find predictive accuracy. The dynamics of the parameters when using the full iterative strategy also requires exploration, with the understanding obtained not specific to any one data set, but generalised beyond that.

Finally, the initial findings have highlighted a potential problem of fuzzy rule induction using iterative rule learning, where individual rules may well classify correctly within their own class, but when combined with the rest of the rule set may result in inferior classification abilities. This suggests that an approach whereby the fuzzy rules are evolved simultaneously, and evaluated together, might yield better results. The authors will be working on extending *FRANTIC* to work in an alternative mode to that of iterative rule learning - by evolving the fuzzy rules of a rule set simultaneously. This may be accomplished by running several ACO algorithms in parallel, with each finding rules for one class.

REFERENCES

[1] P. Kuntz, P. Layzell, and D. Snyers, "A colony of ant-like agents for partitioning in VLSI technology," in *Proc. 4th European Conf. on Artifi cial Life*, Brighton, UK, Jul. 1997, pp. 417–424.

[2] M. Dorigo, V. Maziezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Systems, Man, and Cybernetics B*, vol. 26, pp. 29–41, Feb. 1996.

[3] R. Parpinelli, H. Lopes, and A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 321–332, Aug. 2002.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artifi cial Systems*. New York: Oxford University Press, 1999.

[5] J. Casillas, O. Cordon, and F. Herrera, "Learning fuzzy rules using ant colony optimization algorithms," in *Proc. 2nd International Workshop on Ant Algorithms*, Brussels, Belgium, Sep. 2000, pp. 13–21.

[6] B. Kosko, "Fuzzy entropy and conditioning," *Information Sciences*, vol. 40, pp. 165–174, Dec. 1986.

[7] S-M. Chen, S-H. Lee, and C-H Lee, "A new method for generating fuzzy rules from numerical data for handling classifi cation problems," *Applied Artifi cial Intelligence*, vol. 15, pp. 645–664, Aug. 2001.

[8] Y. Yuan and H. Zhuang, "A genetic algorithm for generating fuzzy classifi cation rules," *Fuzzy Sets and Systems*, vol. 84, pp. 1–19, Nov. 1996.

[9] Y. Yuan and M. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, pp. 125–139, Jan. 1995.